eBird API Tutorial

Eric Nost

January 27, 2024

1 Tutorial: Using the eBird API for Digital Conservation

This tutorial teaches how to access eBird observations data through the citizen science platform's Application Programming Interface (API) and illustrates a very basic use of such data for conservation analysis.

The tutorial is designed for a wide range of users. For those with minimal training in programming, the entire notebook can be "played" without writing a single line of code. For those with experience in the Python programming language, the notebook can be copied and adjusted. Those unfamiliar with "digital conservation" will learn a bit more about an important tool by which species can be tracked, while those already in the conservation sector will have the chance to engage with this familiar dataset in a new way.

1.1 eBird

First, we introduce eBird. eBird is a citizen science platform for bird species tracking and identification. Users report sightings of birds and upload them to the platform, which displays these observations on a map anyone can access. From a user perspective, this provides a digital way birders can manage their checklists. From a researcher and conservationist perspective, eBird provides observational data in volumes that no single team could ever possible hope to acquire. eBird data can help illustrate migration patterns, where rare and threatened species are found, and show us where people are interacting with nature.

1.1.1 eBird's API

An API is a standardized, structured set of names and codes that enable the public to access platform data. Most platforms provide an API, though they differ in how much access they provide. For instance, X (formerly Twitter), has an API that allows researchers to search for tweets based on keywords, geographic location, hashtags, and more and to download these without ever actually opening the website and using its search tool.

Each platform's API is unique and what can be accessed how varies. The only way to know is to review the API documentation. eBird's is available here. These are often not easy to read unless you have a background in programming as well as extensive familiarity with the platform and its data.

For our purposes, we are interested in observations, so we open the `data/obs` folder and select `Recent observations in a region`. This provides us with the general template we need to follow to access relevant eBird data. The query parameters are variables we can adjust in order to access the exact kind of data we want.

In the following cell of code, we import some code that will help us later (`requests`) and then "call up" the eBird API. Basically what we are doing is using programming to "visit" a website, one that contains the data we want, but in the JSON format. To confirm this is what's happening, try entering the url into a browser.

Note: many APIs require you to create an account first and/or get a "token" (password). This may be separate from your platform account. To access eBird's data through its API, you will need to get an API token. You can do so here.

[4]:
```
import requests

url = "https://api.ebird.org/v2/data/obs/KZ/recent"

payload={}
headers = {
  'X-eBirdApiToken': '{{x-ebirdapitoken}}' # Replace {{x-ebirdapitoken}} with␣
 ↪your API token
}

response = requests.request("GET", url, headers=headers, data=payload)

print(response.json()[0:5])
```

```
[{'speciesCode': 'brambl', 'comName': 'Brambling', 'sciName': 'Fringilla
montifringilla', 'locId': 'L16827294', 'locName': '   --
(             ) [Atreyu--TCO Village area (Ural River banks)]',
'obsDt': '2024-01-26 13:59', 'howMany': 9, 'lat': 47.099957, 'lng': 51.9071,
'obsValid': True, 'obsReviewed': False, 'locationPrivate': False, 'subId':
'S159803932'}, {'speciesCode': 'hoocro1', 'comName': 'Hooded Crow', 'sciName':
'Corvus cornix', 'locId': 'L16827294', 'locName': '   --
(             ) [Atreyu--TCO Village area (Ural River banks)]',
'obsDt': '2024-01-26 13:59', 'howMany': 18, 'lat': 47.099957, 'lng': 51.9071,
'obsValid': True, 'obsReviewed': False, 'locationPrivate': False, 'subId':
'S159803932'}, {'speciesCode': 'gretit1', 'comName': 'Great Tit', 'sciName':
'Parus major', 'locId': 'L16827294', 'locName': '   --
(             ) [Atreyu--TCO Village area (Ural River banks)]',
'obsDt': '2024-01-26 13:59', 'howMany': 3, 'lat': 47.099957, 'lng': 51.9071,
'obsValid': True, 'obsReviewed': False, 'locationPrivate': False, 'subId':
'S159803932'}, {'speciesCode': 'fieldf', 'comName': 'Fieldfare', 'sciName':
'Turdus pilaris', 'locId': 'L16827294', 'locName': '   --
(             ) [Atreyu--TCO Village area (Ural River banks)]',
'obsDt': '2024-01-26 13:59', 'howMany': 1, 'lat': 47.099957, 'lng': 51.9071,
'obsValid': True, 'obsReviewed': False, 'locationPrivate': False, 'subId':
'S159803932'}, {'speciesCode': 'eurgre1', 'comName': 'European Greenfinch',
'sciName': 'Chloris chloris', 'locId': 'L16827294', 'locName': '   --
          (             ) [Atreyu--TCO Village area (Ural River
banks)]', 'obsDt': '2024-01-26 13:59', 'howMany': 1, 'lat': 47.099957, 'lng':
51.9071, 'obsValid': True, 'obsReviewed': False, 'locationPrivate': False,
```

```
'subId': 'S159803932'}]
```

The above cell of code returns a list of recent observations in "KZ" (Kazakhstan). You might have to scroll a long ways to see it all! That's because of how the eBird template code works. We can make some adjustments to it.

Below, we change our region to Ontario, Canada (CA-ON). We also format it better using the `pandas` programming package.

[ ]:
```
import json
import pandas

# Here, we create a function that we can reuse later to get different kinds of␣
 ↪data from eBird
def get_ebird_data(url):

  payload={}
  headers = {
    'X-eBirdApiToken': '{{x-ebirdapitoken}}' # Replace {{x-ebirdapitoken}} with␣
 ↪your API token
  }

  response = requests.request("GET", url, headers=headers, data=payload) #␣
 ↪"Visit" the website with this data
  #try:
  #  data = json.loads(response.text) # Load the data in the JSON format
  #except:
  data = pandas.read_json(response.text)

  return data

data = get_ebird_data("https://api.ebird.org/v2/data/obs/CA-ON/recent") #␣
 ↪Change region to CA-ON
data
```

[ ]:

	speciesCode	comName	sciName	locId	\
0	rebwoo	Red-bellied Woodpecker	Melanerpes carolinus	L385611	
1	merlin	Merlin	Falco columbarius	L7944575	
2	whbnut	White-breasted Nuthatch	Sitta carolinensis	L26394771	
3	bkcchi	Black-capped Chickadee	Poecile atricapillus	L26394771	
4	amegfi	American Goldfinch	Spinus tristis	L4140370	
..	…	…	…	…	
192	sprgro	Spruce Grouse	Canachites canadensis	L27290314	
193	baisan	Baird's Sandpiper	Calidris bairdii	L418231	
194	greegr	Great Egret	Ardea alba	L7	
195	leasan	Least Sandpiper	Calidris minutilla	L382375	
196	osprey	Osprey	Pandion haliaetus	L2175806	

```
                                              locName             obsDt  \
0                                   Wildwood Reservoir  2023-12-05 11:55
1    57-299 Ravencrest Rd, Georgina CA-ON (44.2144,…  2023-12-05 11:54
2                                      Heyden, Ontario  2023-12-05 11:52
3                                      Heyden, Ontario  2023-12-05 11:52
4                  Queensville - Eves-Kinsley Residence  2023-12-05 11:50
..                                                  …                 …
192          Snake Falls Rd, Kenora, Unorganized CA-ON  2023-11-24 15:41
193                             Blenheim Sewage Lagoons  2023-11-24 11:24
194                                    Holiday Beach CA  2023-11-24 09:00
195          Harrow Sewage Lagoons (restricted access)  2023-11-23 16:00
196  Burlington--Royal Botanical Gardens (Cherry Hi…  2023-11-23 09:14

     howMany        lat        lng  obsValid  obsReviewed  locationPrivate  \
0        1.0  43.242202 -81.044083      True        False            False
1        1.0  44.214377 -79.400671      True        False             True
2        1.0  46.642828 -84.306234      True        False             True
3        NaN  46.642828 -84.306234      True        False             True
4        5.0  44.136378 -79.449421      True        False             True
..       …        …        …       …           …               …
192      1.0  50.832656 -93.463080      True        False             True
193      1.0  42.319589 -82.020750      True         True            False
194      1.0  42.034026 -83.040726      True         True            False
195      1.0  42.044640 -82.940168      True         True            False
196      1.0  43.294512 -79.878824      True         True            False

          subId exoticCategory
0    S155793499            NaN
1    S155793460            NaN
2    S155793474            NaN
3    S155793474            NaN
4    S155793575            NaN
..          …              …
192  S155126614            NaN
193  S155105425            NaN
194  S155130388            NaN
195  S155101251            NaN
196  S155064199            NaN

[197 rows x 14 columns]
```

1.1.2 API Documentation

At this point, it's worth reviewing the eBird API in a bit more detail to learn more about what exactly we're getting when we make this "request" to the platform API.

The API documentation states that "Results include only the most recent observation for each species in the region specified." So there won't be multiple osprey observations, as an example,

unfortunately. If we wanted to focus on a specific species and ALL recent observations of it, we could, but that would require using a different part of the API. Instead, this allows us to get a general sense of the range of species being observed in the region recently.

Another parameter is `back`, which specifies how far back the API will go into the eBird database to retrieve results. The default is 14 days but we can set it to up to 30 by changing this in our list of `headers`.

[ ]:
```
data = get_ebird_data("https://api.ebird.org/v2/data/obs/CA-ON/recent?back=30")␣
↪# Look back 30 days
data
```

[ ]:
```
    speciesCode                  comName               sciName      locId  \
0        rebwoo   Red-bellied Woodpecker  Melanerpes carolinus    L385611
1        merlin                   Merlin     Falco columbarius   L7944575
2        whbnut  White-breasted Nuthatch    Sitta carolinensis  L26394771
3        bkcchi   Black-capped Chickadee  Poecile atricapillus  L26394771
4        eursta         European Starling      Sturnus vulgaris   L4140370
..          …                       …                     …        …
250      eargre              Eared Grebe  Podiceps nigricollis   L2891330
251      limpki                  Limpkin       Aramus guarauna  L27844578
252      chukar                   Chukar      Alectoris chukar  L12821276
253      pinwar             Pine Warbler       Setophaga pinus    L131153
254      indbun           Indigo Bunting      Passerina cyanea    L131153

                                                locName              obsDt  \
0                                    Wildwood Reservoir  2023-12-05 11:55
1    57-299 Ravencrest Rd, Georgina CA-ON (44.2144,…  2023-12-05 11:54
2                                       Heyden, Ontario  2023-12-05 11:52
3                                       Heyden, Ontario  2023-12-05 11:52
4                  Queensville - Eves-Kinsley Residence  2023-12-05 11:50
..                                                  …                …
250                                  Ipperwash Beach  2023-11-07 07:32
251                              Watson's Corners Road  2023-11-06 15:15
252                                          Sufian St  2023-11-06 14:01
253  Point Pelee National Park (general location fo…  2023-11-06 06:53
254  Point Pelee National Park (general location fo…  2023-11-06 06:53

     howMany        lat        lng  obsValid  obsReviewed  locationPrivate  \
0        1.0  43.242202 -81.044083      True        False            False
1        1.0  44.214377 -79.400671      True        False             True
2        1.0  46.642828 -84.306234      True        False             True
3        NaN  46.642828 -84.306234      True        False             True
4        8.0  44.136378 -79.449421      True        False             True
..       …        …        …       …          …                …
250      1.0  43.206677 -81.995859      True         True            False
251      1.0  44.975124 -76.536940      True         True             True
252      1.0  45.819281 -77.220064      True         True             True
```

```
253      1.0  41.955399 -82.514000      True         True            False
254      1.0  41.955399 -82.514000      True         True            False

          subId exoticCategory
0    S155793499            NaN
1    S155793460            NaN
2    S155793474            NaN
3    S155793474            NaN
4    S155793575              N
..          …            …
250  S153968504            NaN
251  S153926361            NaN
252  S154267581              X
253  S153897439            NaN
254  S153897439            NaN

[255 rows x 14 columns]
```

1.1.3 Attribute Analysis

Now that we have a solid set of results, we can proceed to analyze them towards the goal of understanding the range of species observed in Ontario in the past 30 days - and where they were observed.

First, we'll use the `pandas` programming package to examine key attributes of these observations. Then, we'll use `geopandas` - it's spatial data equivalent - to map them and assess their spatial distribution.

The following cell of code transforms our JSON-formatted data into a nicely formatted table.

[ ]:
```
import pandas

data = pandas.DataFrame(data)

data
```

[ ]:
```
    speciesCode                 comName              sciName      locId  \
0        rebwoo   Red-bellied Woodpecker  Melanerpes carolinus    L385611
1        merlin                  Merlin    Falco columbarius   L7944575
2        whbnut  White-breasted Nuthatch   Sitta carolinensis  L26394771
3        bkcchi   Black-capped Chickadee  Poecile atricapillus  L26394771
4        eursta         European Starling     Sturnus vulgaris   L4140370
..          …                       …                  …         …
250      eargre              Eared Grebe  Podiceps nigricollis   L2891330
251      limpki                  Limpkin      Aramus guarauna  L27844578
252      chukar                   Chukar     Alectoris chukar  L12821276
253      pinwar             Pine Warbler       Setophaga pinus    L131153
254      indbun           Indigo Bunting     Passerina cyanea    L131153
```

```
                                                locName              obsDt  \
0                                     Wildwood Reservoir  2023-12-05 11:55
1    57-299 Ravencrest Rd, Georgina CA-ON (44.2144,…  2023-12-05 11:54
2                                        Heyden, Ontario  2023-12-05 11:52
3                                        Heyden, Ontario  2023-12-05 11:52
4                    Queensville - Eves-Kinsley Residence  2023-12-05 11:50
..                                                    …                …
250                                   Ipperwash Beach  2023-11-07 07:32
251                              Watson's Corners Road  2023-11-06 15:15
252                                         Sufian St  2023-11-06 14:01
253  Point Pelee National Park (general location fo…  2023-11-06 06:53
254  Point Pelee National Park (general location fo…  2023-11-06 06:53

     howMany        lat        lng  obsValid  obsReviewed  locationPrivate  \
0        1.0  43.242202 -81.044083      True        False            False
1        1.0  44.214377 -79.400671      True        False             True
2        1.0  46.642828 -84.306234      True        False             True
3        NaN  46.642828 -84.306234      True        False             True
4        8.0  44.136378 -79.449421      True        False             True
..       …        …        …       …           …               …
250      1.0  43.206677 -81.995859      True         True            False
251      1.0  44.975124 -76.536940      True         True             True
252      1.0  45.819281 -77.220064      True         True             True
253      1.0  41.955399 -82.514000      True         True            False
254      1.0  41.955399 -82.514000      True         True            False

          subId exoticCategory
0    S155793499            NaN
1    S155793460            NaN
2    S155793474            NaN
3    S155793474            NaN
4    S155793575              N
..          …              …
250  S153968504            NaN
251  S153926361            NaN
252  S154267581              X
253  S153897439            NaN
254  S153897439            NaN

[255 rows x 14 columns]
```

Next, we analyze the `sciName` column to figure out exactly how many different species were observed.

[ ]:
```
data["sciName"].nunique()
```

[ ]: 255

So there were 255 different species identified over the past 30 days in southern Ontario (this number will vary depending on when this tutorial is run!)

What kinds of species were these? This will require us to make a different, separate call to eBird platform. This is not unusual - APIs typically provide very segmented information, requiring multiple calls to get the data you want. First, we will acquire more information for each species observed.

[ ]:
```
observed = list(data["speciesCode"].unique()) # A list of each unique species␣
 ↪observed and their eBird species code
observed_formatted = ""
for bird in observed:
  observed_formatted += bird+","
observed_formatted = observed_formatted[:-1]
# In the above lines of code, we took our list of unique species observed and␣
 ↪formatted it in the way the eBird API requires (a comma-separated list)

species_information = get_ebird_data("https://api.ebird.org/v2/ref/taxonomy/
 ↪ebird?fmt=json&species="+observed_formatted)
species_information
```

[ ]:

	sciName	comName	speciesCode	\
0	Anser caerulescens	Snow Goose	snogoo	
1	Anser rossii	Ross's Goose	rosgoo	
2	Anser caerulescens x rossii	Snow x Ross's Goose (hybrid)	sxrgoo1	
3	Anser anser	Graylag Goose	gragoo	
4	Anser albifrons	Greater White-fronted Goose	gwfgoo	
..	…	…	…	
250	Piranga rubra	Summer Tanager	sumtan	
251	Cardinalis cardinalis	Northern Cardinal	norcar	
252	Pheucticus ludovicianus	Rose-breasted Grosbeak	robgro	
253	Passerina cyanea	Indigo Bunting	indbun	
254	Spiza americana	Dickcissel	dickci	

	category	taxonOrder	bandingCodes	comNameCodes	sciNameCodes	\
0	species	256	[SNGO]	[]	[ANCA]	
1	species	260	[ROGO]	[]	[ANRO]	
2	hybrid	261	[SRGH]	[SNGO, ROGO]	[ANRO, ANCA]	
3	species	263	[GRGO]	[]	[ANAN]	
4	species	271	[GWFG]	[]	[ANAL]	
..	…	…	…	…	…	
250	species	33885	[SUTA]	[]	[PIRU]	
251	species	33967	[NOCA]	[]	[CACA]	
252	species	34009	[RBGR]	[]	[PHLU]	
253	species	34060	[INBU]	[]	[PACY]	
254	species	34080	[DICK]	[]	[SPAM]	

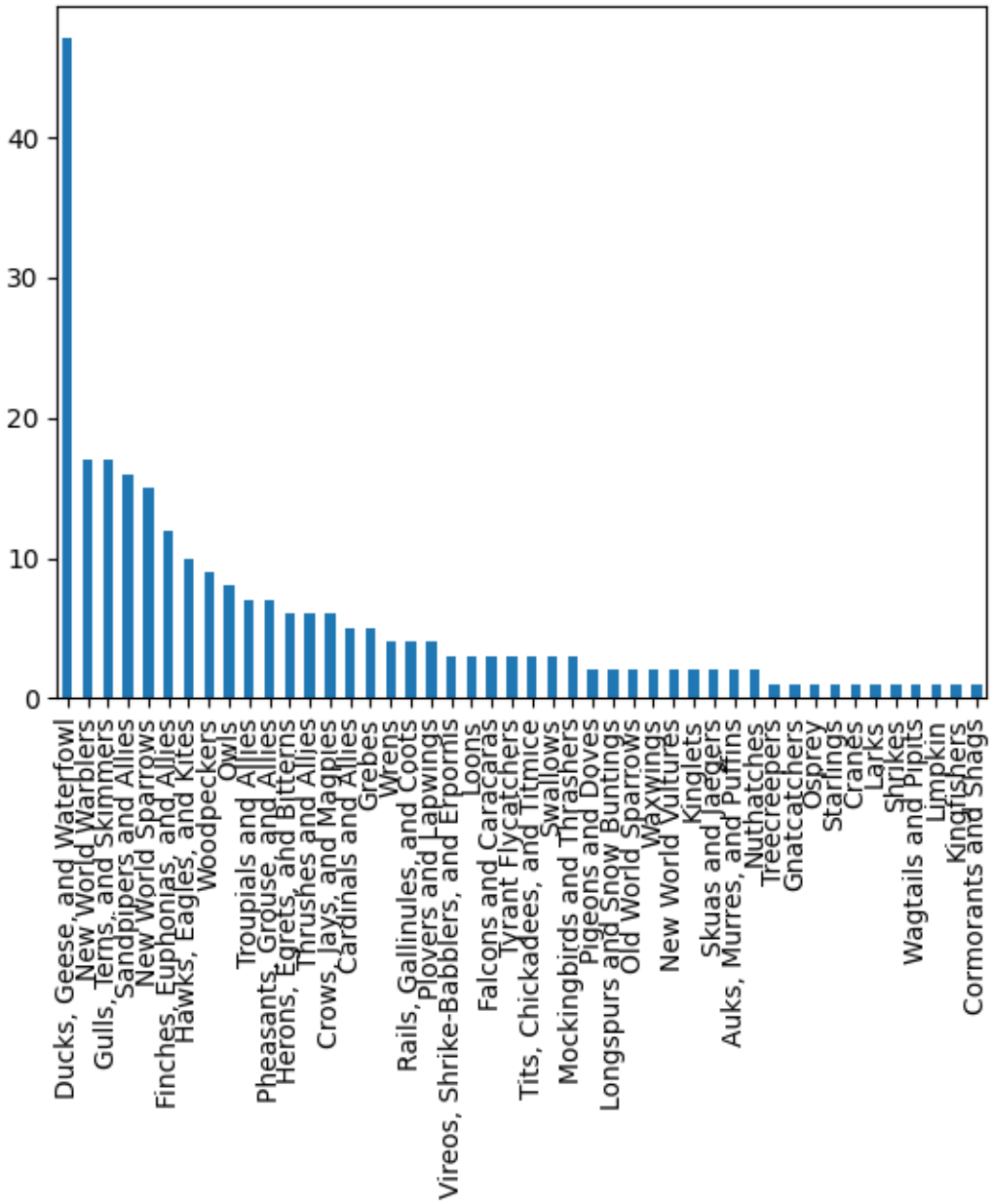
```
             order familyCode                familyComName familySciName
0      Anseriformes    anatid1  Ducks, Geese, and Waterfowl      Anatidae
1      Anseriformes    anatid1  Ducks, Geese, and Waterfowl      Anatidae
2      Anseriformes    anatid1  Ducks, Geese, and Waterfowl      Anatidae
3      Anseriformes    anatid1  Ducks, Geese, and Waterfowl      Anatidae
4      Anseriformes    anatid1  Ducks, Geese, and Waterfowl      Anatidae
..              …          …                          …           …
250  Passeriformes    cardin1         Cardinals and Allies  Cardinalidae
251  Passeriformes    cardin1         Cardinals and Allies  Cardinalidae
252  Passeriformes    cardin1         Cardinals and Allies  Cardinalidae
253  Passeriformes    cardin1         Cardinals and Allies  Cardinalidae
254  Passeriformes    cardin1         Cardinals and Allies  Cardinalidae

[255 rows x 12 columns]
```

We can summarize this information and then plot it in a histogram.

[ ]:
```
species_information['familyComName'].value_counts().plot(kind='bar')
```

[ ]:
```
<Axes: >
```



To no one's surprise, we're mostly observing ducks, geese, and waterfowl!

Now let's see how representative this list of species is compared to ALL species ever observed in Ontario. We'll get that information from yet another part of eBird's API - that is, using another, different URL.

[ ]:
```
species_list = get_ebird_data("https://api.ebird.org/v2/product/spplist/CA-ON")
species_list
```

[ ]:
```
          0
0    bbwduc
1    fuwduc
2    bahgoo
3    empgoo
4    snogoo
..      …
639  lazbun
640  indbun
641  varbun
642  paibun
643  dickci

[644 rows x 1 columns]
```

There have been many species observed over time in Ontario. In the past 30 days, we have observed the following percent of these species:

[ ]:
```
str(round((species_information['speciesCode'].nunique()/species_list[0].
 ↪shape[0]) * 100, 2)) + "%"
```

[ ]:
```
'39.6%'
```

1.1.4 Spatial Analysis

Finally, let's look at WHERE these species were observed. Because of the nature of the eBird API, we don't have access to ALL observations in the recent past, so we can't really make claims about the spatial distribution of bird sightings (you would need to manually download the data from the Global Biodiversity Information Facility to do that). Nonetheless, we could consider what we do have - the most recent observation of each species observed in the past 30 days in Ontario - as perhaps representative of where citizen scientists are operating. However, it is crucial to remember that these data reflect where people who use eBird are finding birds. The data tell us as much about people as birds. We are likely to map eBird observations near cities, in parks, and so on.

First, we'll load the `geopandas` and `folium` programming packages then we'll load our data into the geopandas format and use folium to display them in an interactive map.

[ ]:
```
import geopandas
import folium

map_data = geopandas.GeoDataFrame(data, geometry=geopandas.
 ↪points_from_xy(data['lng'], data['lat']), crs = 4326)

m = folium.Map()

birds_layer = folium.GeoJson(
    data = map_data,
```

```
    tooltip = folium.GeoJsonTooltip(fields=["comName", "sciName", "obsDt",␣
  ↪"howMany"])
).add_to(m)

bounds = m.get_bounds()
m.fit_bounds(bounds, padding=0)

m
```

[ ]: <folium.folium.Map at 0x7af56c5e93f0>

You may not be surprised to hear at this point that we can use another part of eBird's API to get around this limitation to accessing bird observation data. While we may not be able to get all observations for all species all at once, we can use the "Recent observations of a species in a region" query to acquire a more robust sample of observations in an area for a single species, and by calling this query more than once, we can gather data for more than one species.

It is important to note that, per the eBird API documentation, we will still only be able to get the most recent observation made in each "location" in the region. What eBird considers a "location" is unclear. It uses various "codes" to describe locations. For instance, L227544 refers to a specific area near Cornell University. For our purposes, this just means that we will be sampling locations in the Ontario region with observations of a species or two in the past 30 days, but we will not be able to say anything specific about how many total observations were made.

So, let's compare where two random species were observed. Keep in mind that there may not be that many observations of relatively rare species.

[ ]:
```
# First, get two random species observed in Ontario in the past 30 days
random = list(data.sample(2)["speciesCode"].unique())

# Get more observations of these
observations = pandas.DataFrame()
for species in random:
  species_data = get_ebird_data("https://api.ebird.org/v2/data/obs/CA-ON/recent/
  ↪"+species+"?back=30")
  observations = pandas.concat([observations, species_data])

# Map the observations
## Different colours will indicate different species, and different sizes of␣
  ↪circles will reflect how many individual birds were observed in that␣
  ↪location at that time
colormap = {random[0]: "orange", random[1]: "blue"}
def size(cnt):
  radius = 1
  if cnt < 10:
    radius = 8
  elif cnt < 20:
    radius = 12
```

```
  elif cnt < 30:
    radius = 16
  elif cnt < 50:
    radius = 20
  else:
    radius = 32
  return radius

m = folium.Map()

birds = [folium.CircleMarker(location=[bird["lat"], bird["lng"]],
  popup=folium.Popup(bird["comName"]+'<br><b>Scientific Name:</b>␣
 ↪'+bird["sciName"]+'<br><b>Location:</b> '+bird["locName"]+'<br><b>Date most␣
 ↪recently observed:</b> '+bird["obsDt"]),
  radius=size(bird["howMany"]), fill_color=colormap[bird["speciesCode"]],␣
 ↪color="black", weight=1, fill_opacity = 1) for index,bird in observations.
 ↪iterrows()]

for bird in birds:
  m.add_child(bird)

bounds = m.get_bounds()
m.fit_bounds(bounds, padding=0)

m
```

[ ]:
```
<folium.folium.Map at 0x7af56eea7e80>
```

[ ]: